Welcome to Kid Krypto
An introduction to cryptography

User Guide

5/4/07

SD0617
Ben Anderson
Deepak Agarwal
Advisor: Dr. Raj Katti

Introduction: What is Cryptograpy?

Cryptography, from the Greek, literally means "hidden-write." Cryptography attempts to satisfy the requirements of information security by making a communicated message unreadable to any party outside the sender and receiver. However, this only scratches the surface of everything information security entails. For our introductory purposes, this definition will suffice.

There are many methods of encrypting, or hiding, a message from prying eyes. Some range from very simple substitution methods to very complex mathematical formula-based codes that are impossible to break by conventional methods. Possibly the one true revolution in the history of cryptography occurred with the advent of the public and private key system. Cryptography finally moved from a symmetric substitution and permutation method to a much more difficult asymmetric, mathematical function based scenario. This has had profound effects on security and confidentiality for senders and receivers.

The two-key system we focus on in this tutorial is the RSA algorithm. RSA (named for its three authors, Ron Rivest, Adi Shamir, and Len Adelman) was developed at MIT in 1978 and has become the most widely accepted and used general purpose approach to public and private key encryption systems. The RSA algorithm falls under the category of a trapdoor function. A trapdoor function is one that is easily set up in one direction, but exceptionally difficult to return to that original state without knowledge of the original variables. We will discuss this in more detail soon.

```
     As you progress through this tutorial, please advance
through the program on the Kid Krypto system by pressing any key.
```

Key Generation

Step 1: Choosing Primes

RSA Algorithm: Cryptotext = Message^{e} mod n

At the foundation of the RSA algorithm is a very large number n. The size of n is typically at least 1024 bits, or 309 digits. This can be known as the algorithm's encryption level. In practice, two very large prime numbers (of at least 75-100 digits) are randomly selected and multiplied together to produce n. Our system cannot handle numbers larger than 4.3 billion (10 digits) so our n will be much smaller.

```
Advance the Kid Krypto system until you are asked to enter
prime number p.

Enter a two digit prime number (11,13,17,19,23,29,etc.) and
press the enter key. (If you make a mistake, press the clear key
to clear the form and enter a new number.)

Enter another two digit prime number q, different from the
first and press the enter key.
```

We now have our value n, though not a particularly large value and nowhere near the 309 digit number we seek for *basic* security. This begs the question, how would we find n in real life?

Currently, there are no techniques to randomly generate prime numbers of a determined size. When the prime numbers we are looking for are in the order of 75+ digits, how can we even know if they are prime in the first place? This leads us to our first algorithm examination: the Miller-Rabin algorithm for testing primality.

Step 1.2: The Miller-Rabin Algorithm

The Miller-Rabin algorithm uses **Fermat's Theorem:**

If p is prime and a is a positive integer not divisible by p, then:

$$a^{p-1} = 1 \bmod p$$

or another useful form:

$$a^{p} = a \bmod p$$

The mod function, short for the modulo function, is also known as the remainder function (or residue). In other words,

a mod p is defined as the remainder when a is divided by p.

For example:

$a = 7$ $p = 5$

a mod $p = 7/5$ => 1 remainder 2

a mod $p = 2$

An interesting side venture is determining the formula for the mod function, but we'll leave that discovery up to the reader.

Surprisingly, the Millier-Rabin algorithm does not necessarily yield a prime number, but like almost all other primality tests, the yielded number is *almost* certainly a prime number due to the fact that this algorithm works off of probabilities.

How it works is like follows:

1.) Find integers k, q with $k > 0$, q odd, so that (n-1 = $2^k q$)
2.) Select a random integer a, $1 < a < n - 1$
3.) If a^q mod $n = 1$ then our test is inconclusive (possibly prime)
4.) For all a's 0 to k-1 check that a^{2jq} mod $n = n$-1. (still possibly prime)
5.) If at any time a^{2jq} mod n is not equal to 1 or n-1, then it is not prime.

For Example:

If we were to test if 15 were prime, we would find

$n = 15$ $\quad n - 1 = 14$ $\quad 14 = 2^k q \Rightarrow (2^1)*7$

$k = 1$ $\quad q = 7$

We pick $a = 4$. Then we have

$4^7 \bmod 15 = 4$ $\quad \Rightarrow$ NOT Prime

The 4 we found is neither 1 nor 14, which shows that 15 is not prime.

However, if we would have taken $a = 14$, our equation would be

$14^7 \bmod 15 = 14$ $\quad \Rightarrow$ POSSIBLY Prime

The 14 returned indicates that 15 is possibly prime. For all other values of *a* (0-13) we find that 15 is not prime, and it only takes one test of 15 not being prime to show that it is not prime. So, if we were to only test 15 once, we would have a 7.1% chance of returning 15 as a prime. The more tests there are with different values of *a*, the more drastically that percentage is reduced. In the case of $n = 15$, two tests using two different values of *a* would leave no chance of error. For larger values *n* that aren't prime, there are more chances that *n* could be shown as prime. For instance 13 * 17 = 221, and there are six values of *a* that would show 221 is possibly prime. At some point, there is no way you can check every value of *a* and have to set how many tests you would like to make. A sufficient number of tests that do not return *n* as a possible prime number will *almost* certainly show that *n* is a prime number.

In practice, 75+ digits *p* & *q* are randomly chosen and tested for primality. This can take a long time, but a public and private key only need to be generated once every so often, so the time factor is acceptable. Once they are found, they are multiplied to find *n*.

Step 2: Determine the Secret Message

RSA Algorithm: Cryptotext = Message^{e} mod n

The message, or plaintext, we want to send is a value less than n. As you can see, our message is exponentially taken to a power. This indicates that Message^{e} evaluates to a very large number. To keep the numbers smaller and more meaningful, we will only be taking a message from 2-9.

Scroll through the messages until you are asked to enter a secret message. Press a number between 2 & 9 and press enter. Clear will reset the entry if you make a mistake.

Realistically, as long as the plaintext Message is less than n, the Cryptotext can be calculated.

Step 3: Calculate the Euler Totient

RSA Algorithm: Cryptotext = Message^{e} mod n

The RSA Algorithm uses a relationship that is described by a corollary of Euler's theorem:

Given two prime numbers p and q
Given two integers $n = pq$ and $m < n$
Given an arbitrary value k

$$m^{k(p-1)(q-1)+1} = m \bmod n$$

Phi (or Φ in Greek notation) is used to represent $(p - 1)(q - 1)$, which is also known as the Euler Totient.

$$\Phi(n) = (p - 1)(q - 1)$$

The Euler Totient represents the number of positive integers less than n and that n cannot be divided evenly by. (How many numbers leave remainders when n is divided by them.) We can break Euler's equation down to the following:

$$d = e^{-1} \bmod \Phi(n)$$

Where e and d are multiplicative inverses of each other AND relatively prime to $\Phi(n)$. (Their greatest common divisor is 1.)

Phi is calculated automatically on the Kid Krypto System, but you need to remember this number for the next step!

Step 3.2: Choose *e*

RSA Algorithm: Cryptotext = Message^{e} mod n

Euler's Corollary: $d = e^{-1} \bmod \Phi(n)$

Generally, e is a standard large prime number that is relatively prime to $\Phi(n)$ and less than $\Phi(n)$. In our case, we want to choose e low to keep number sizes manageable. Remember your number phi? We need it to choose an appropriate value e. We need e to be relatively prime to phi, so choose e so that phi does NOT divide evenly by it.

Scroll through the messages until you are asked to enter a value e. Five generally works unless phi ended with a 5 or 0.

```
The system will stop and need to be reset if phi is divisible by
e, so choose carefully!
```

Step 3.3: Calculate the Multiplicative Inverse

RSA Algorithm: Cryptotext = Message^{e} mod n

Euler's Corollary: $d = e^{-1} \bmod \Phi(n)$

When we say $d = e^{-1} \bmod \Phi(n)$, it can be mistaken that e^{-1} is just the inverse of e. What we want is the *multiplicative* inverse, which we can find using an extension of Euclid's algorithm. Euclid's algorithm is a simple procedure for determining the greatest common divisor of two positive integers.

First, we take two positive integers A and B.

1.) Set a <= A and b <= B.
2.) If B = 0, then A is the greatest common denominator of a and b.
3.) Otherwise, find R = A mod B
4.) A <= B
5.) B <= R
6.) goto 2

For Example:

Find the gcd(36, 24)

R = 36 mod 24 = 12	=>	A = 24	B = 12
R = 24 mod 12 = 0	=>	A = 12	B = 0

The greatest common denominator of 36 and 24 is 12.

This formula can be extended to show multiplicative inverses of numbers using the modulo function.

If gcd(m,b) = 1, then b has a multiplicative inverse modulo m. That is, for positive integer $b < m$, there exists a $b^{-1} < m$ such that $bb^{-1} = 1 \bmod m$.

For smaller values, we can use a chart like the following:

Table 1: Multiplication modulo 5

x	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

In the shaded area, we can see that these are the modulo 5 values when the edges are multiplied. Where the modulo equals 1, we can see that the column and row are multiplicative inverses for 5. While this works for smaller values, larger values require an algorithm to calculate. The Kid Krypto System will calculate this value for us.

Advance through the program until d, the multiplicative inverse of e, is calculated. As demonstrated in the chart, not all values less than $\Phi(n)$ have multiplicative inverses. If our value e does not have an inverse, the program will stall. Please restart the program and try again if this happens!

Step 4: The Result

The results we have from the numbers we have chosen and have generated are our public and private keys. The public key is our value *n* that we found from multiplying our chosen primes and the chosen value *e*. The private key is our value *n* again and the multiplicative inverse of *e, d*.

Public Key: [*e,n*] => This is the key we make available to everyone.
Private Key: [*d,n*] => This is the key we keep all to ourselves.

Again, since finding two large random prime numbers takes a bit of computation time, the public and private keys can take some time to generate. Once we have our keys, though, we can reuse them for our secret messages as long as we like.

Encryption and Decryption

Step 1: Pubic Key Encryption

RSA Algorithm: Cryptotext = Message^{e} mod n

For a Message less than n, we can encrypt the message using the RSA Algorithm. It is a simple matter of using the equation with our public key. Since we know our message, e, and n, we just fill in the blanks to obtain the cryptotext. The Kid Krypto System will calculate this value for us.

`Advance through the program until the display shows the encrypted version of the message.`

Step 2: Decryption Using the Private Key

RSA Algorithm^{-1}: Message = Cryptotext^{d} mod n

Decryption is a simple matter of using our generated private key in a very similar way to how we created the cryptotext.